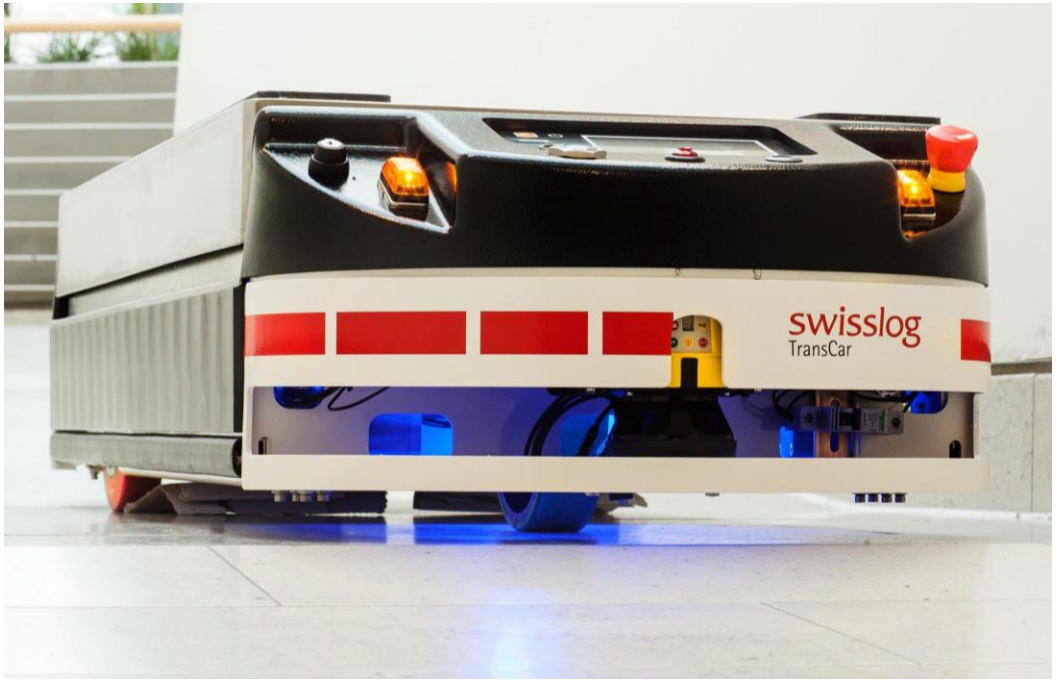
LAISSUS
Aymeric



TIPE – Optimisation du système de transport du matériel au CHU de Bourgogne

Comment optimiser le système de transport en analysant les statistiques fournies par le CHU?

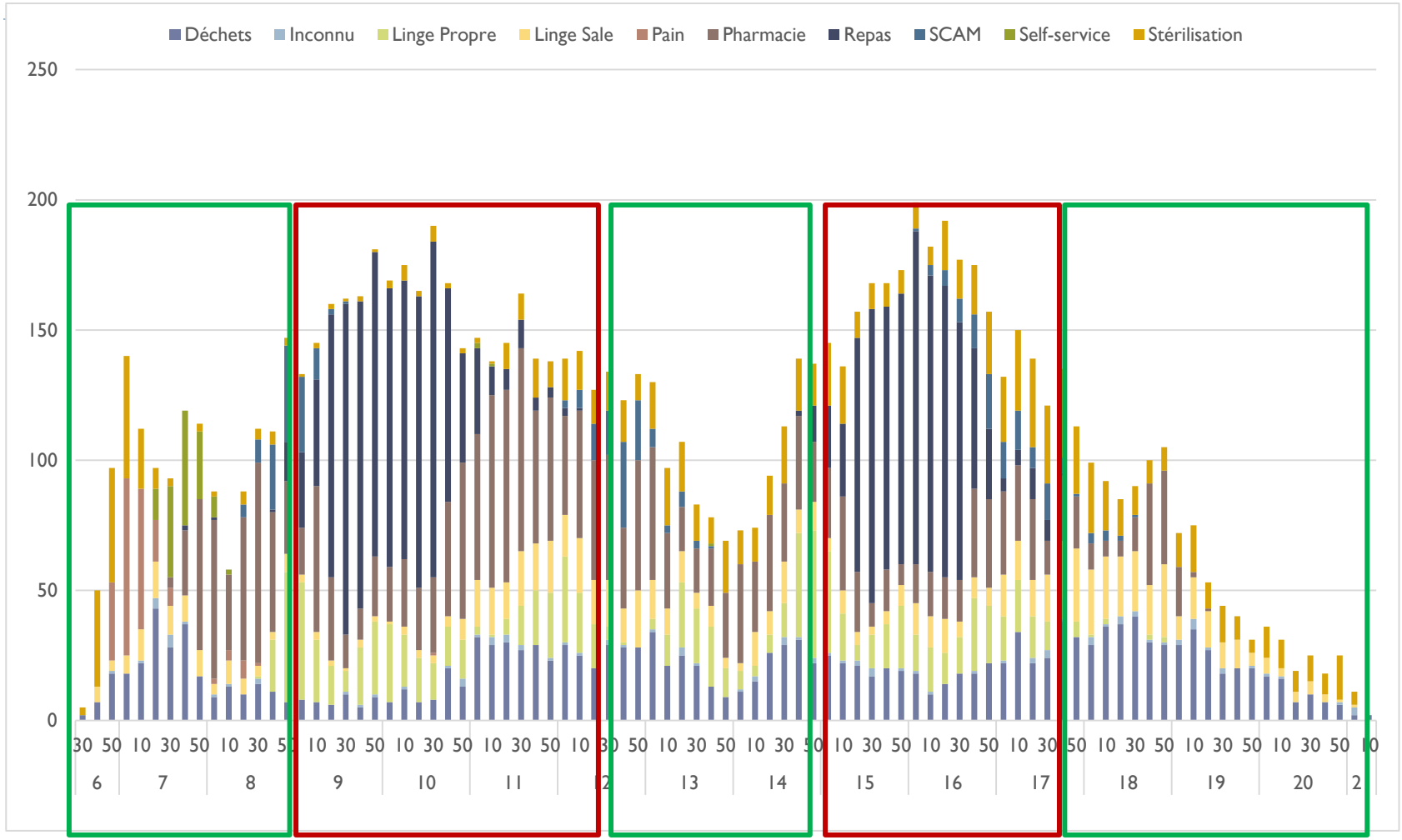
Contexte

- Présentation générale

Centre Hospitalier Universitaire
Dijon Bourgogne

swisslog

Problématique posée

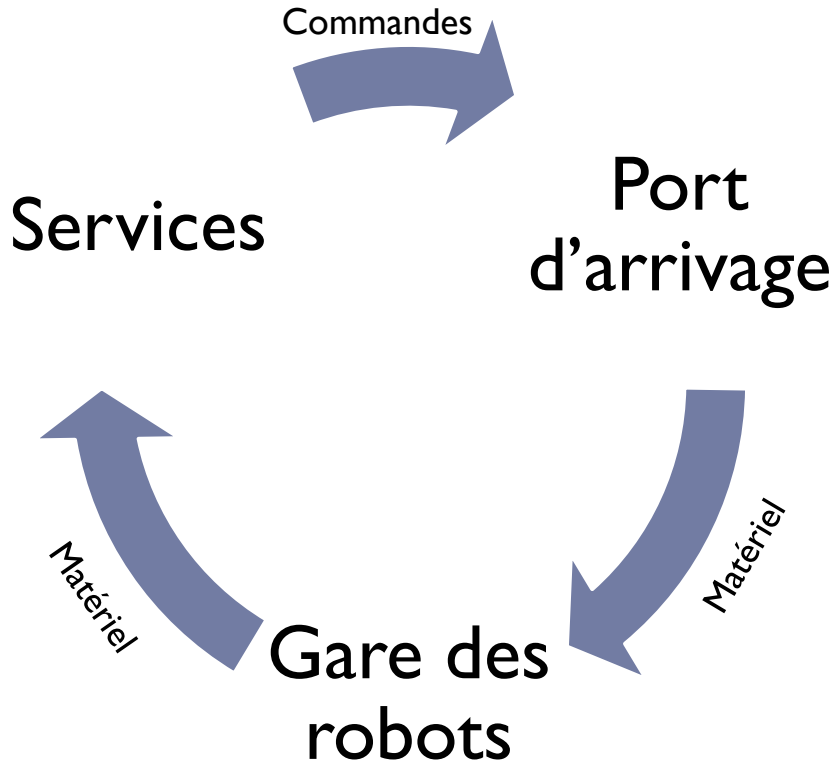


Répartition inégale très visible grâce aux statistiques fournies

Plan de la réflexion

- I. Etude de la chaine d'acheminement et de ses problèmes
- II. Réflexion sur les différentes solutions
- III. Mise en œuvre pratique de la solution choisie
- IV. Conclusion

I. Etude globale de la chaine

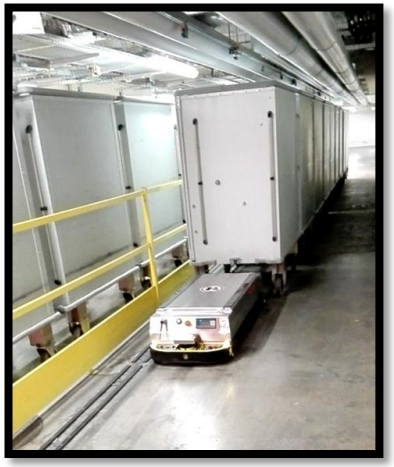


- A. Découverte des différentes étapes
- B. Conclusion sur les problèmes solvables ou non

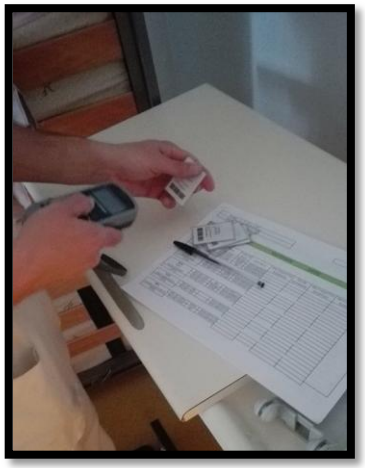
A. Les étapes d'acheminement

- Port d'arrivage

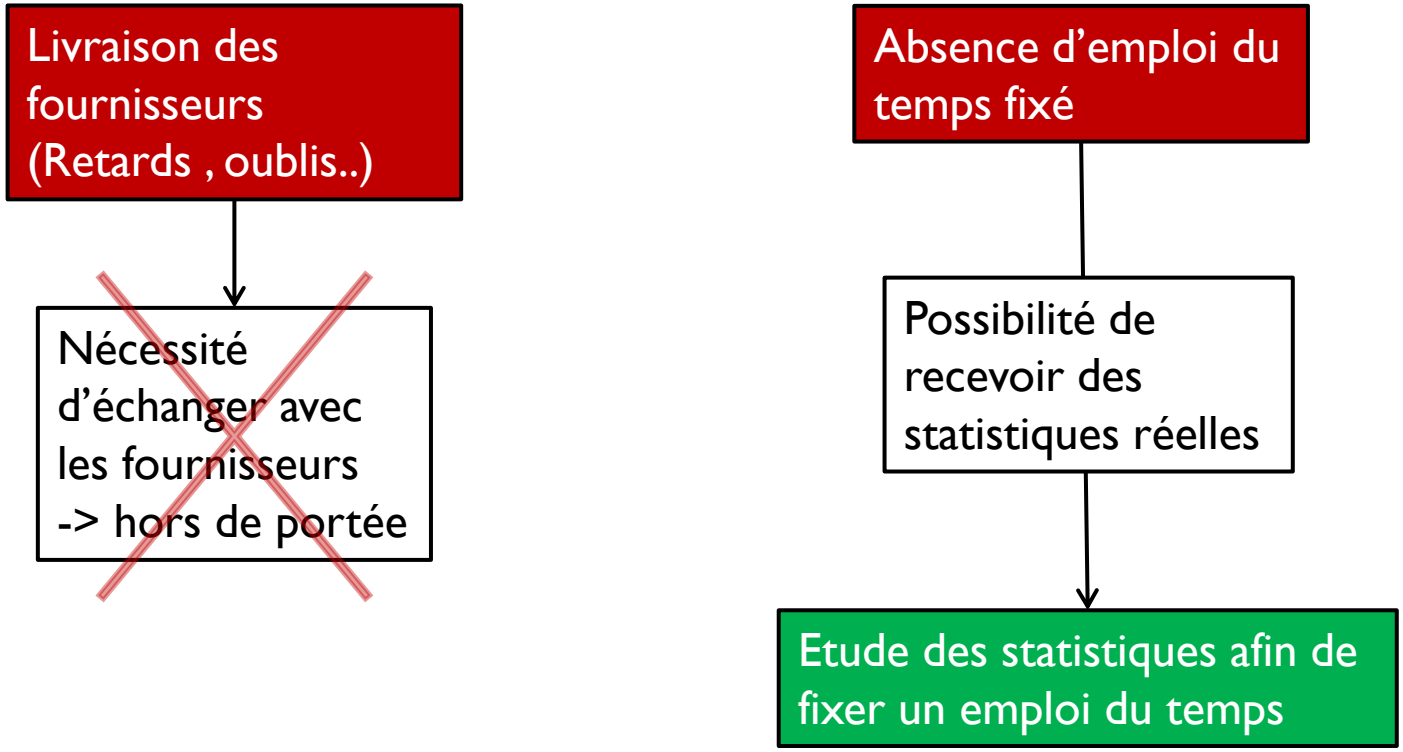
- Gare des robots



- Services

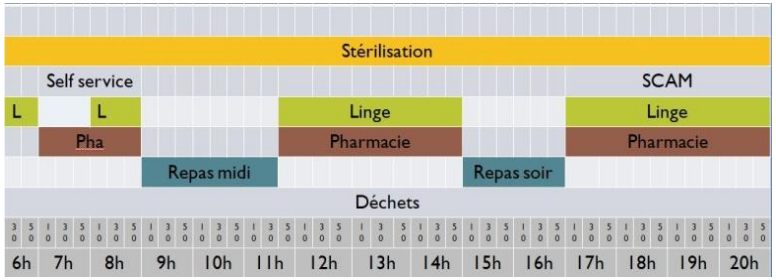


B. Conclusion intermédiaire

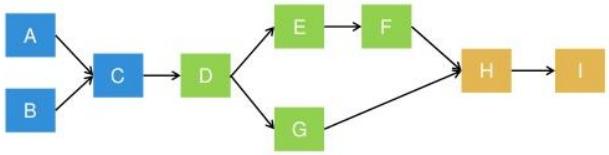


II. Différentes solutions possibles

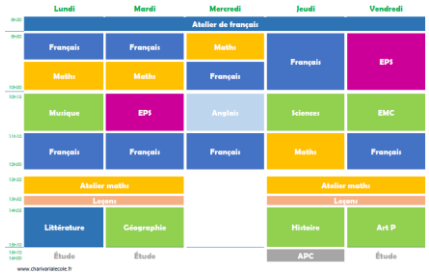
- Solution manuelle



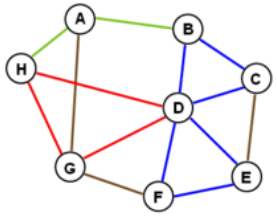
- Solutions algorithmiques



Ordonnancement de tâches



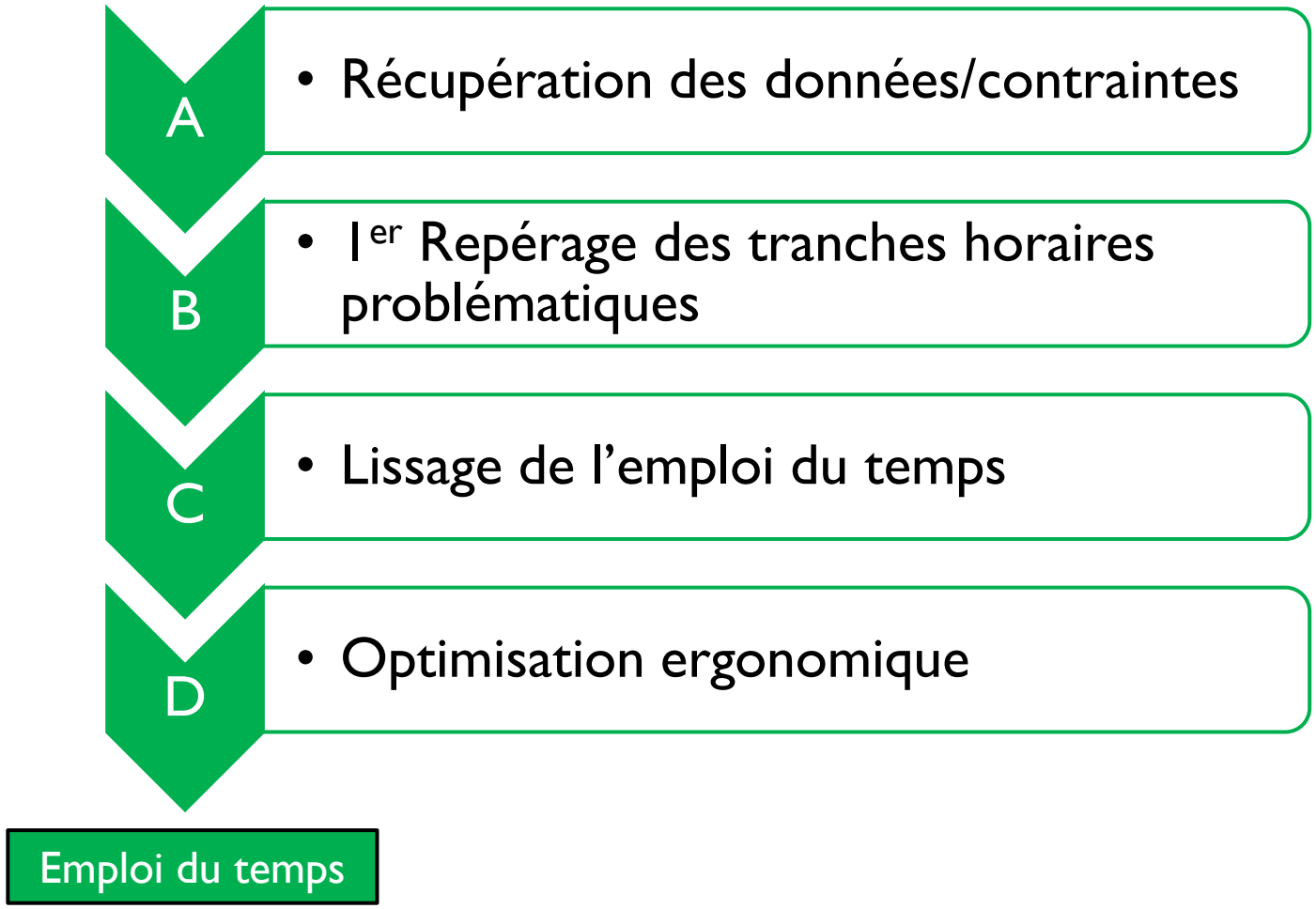
PPC



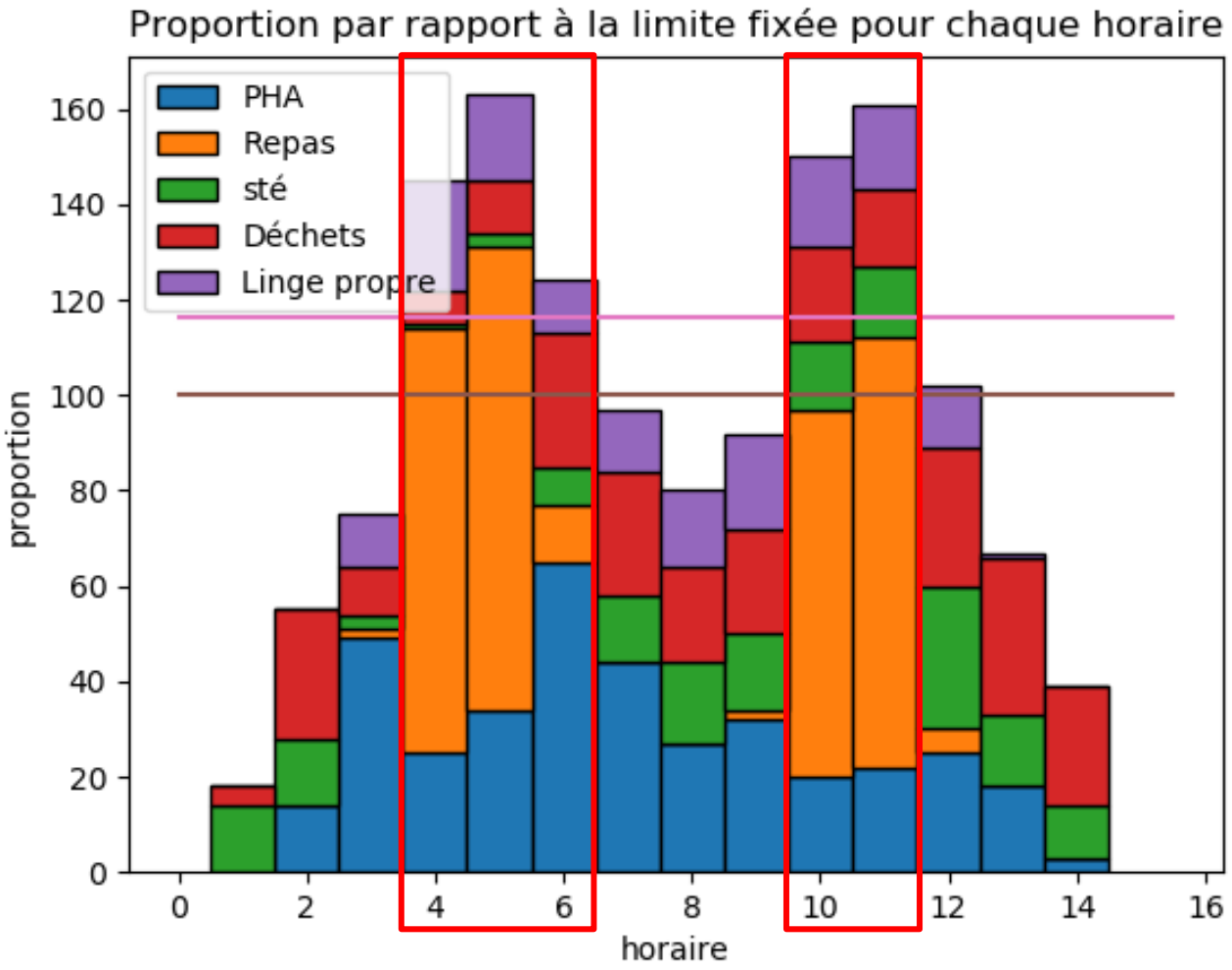
Théorie des graphes

Besoin de créer totalement un algorithme adapté au système

III. Mise en œuvre algorithmique

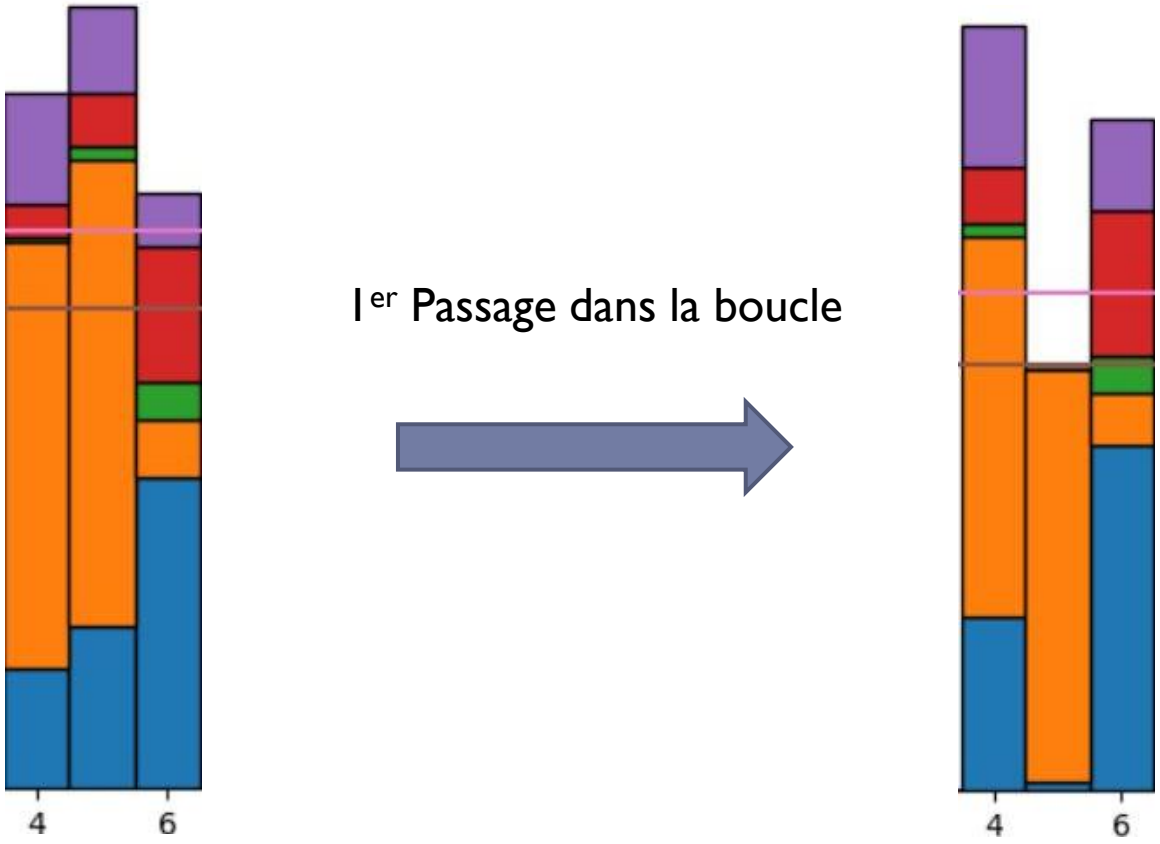


- A. Récupération des données et contraintes
- B. 1er Repérage des tranches horaires problématiques

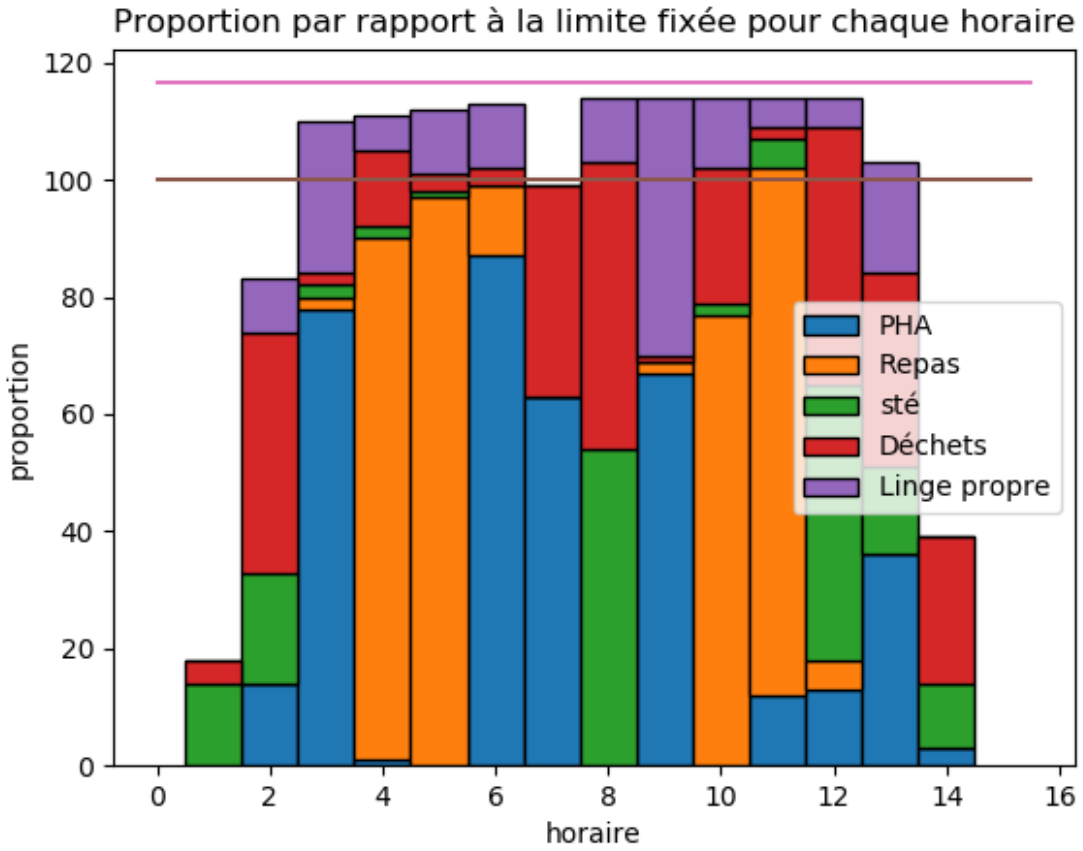


C. Lissage algorithmique de l'emploi du temps

1. Fonctionnement



➢ 2. Résultat final



➢ Respect de la limite fixée

➢ Respect de la contrainte

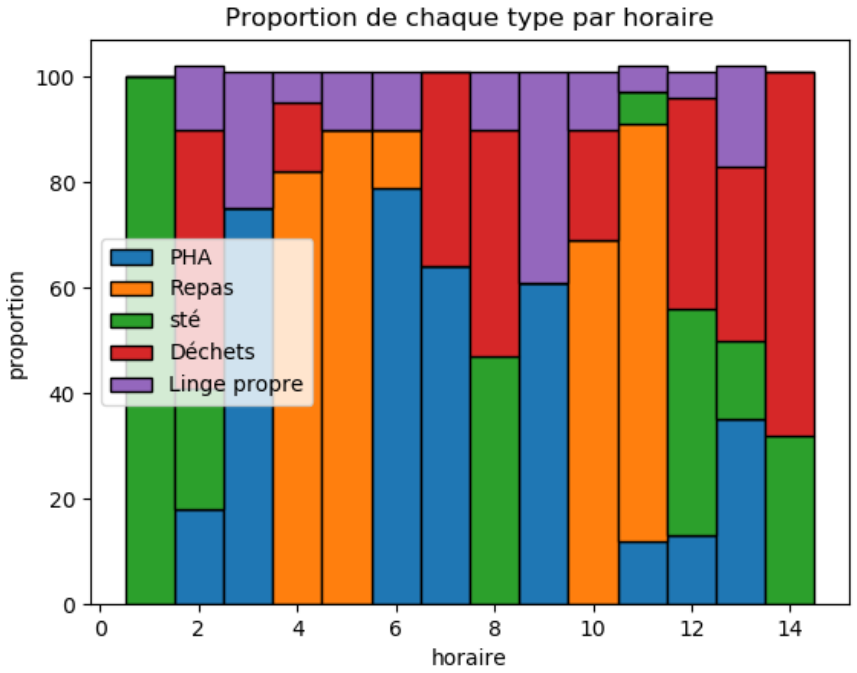
➢ Mais complexe à utiliser

D. Optimisation ergonomique

- Omission des proportions < 3% du total de conteneur par type

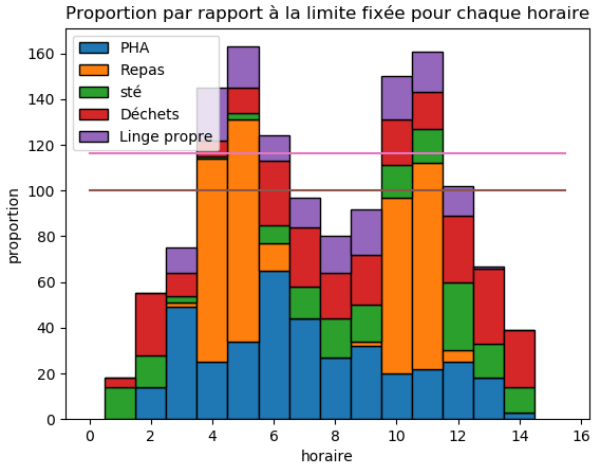
Pertes de conteneurs	
PHA	1%
Repas	2%
Déchets	5%
Sté	5%
Linge propre	0%

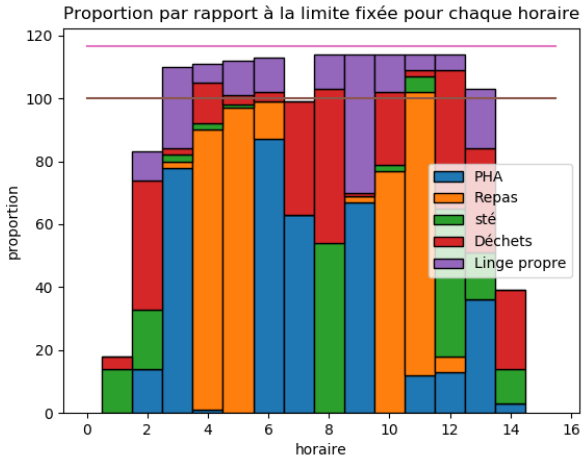
- Visualisation graphique de EDTS
  - Plus clair
  - Illustre mieux les proportions à adopter

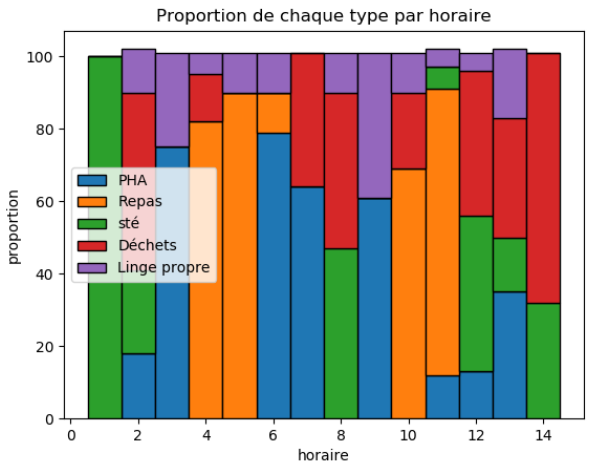


IV. Conclusion

A. Bilan de performances algorithmiques



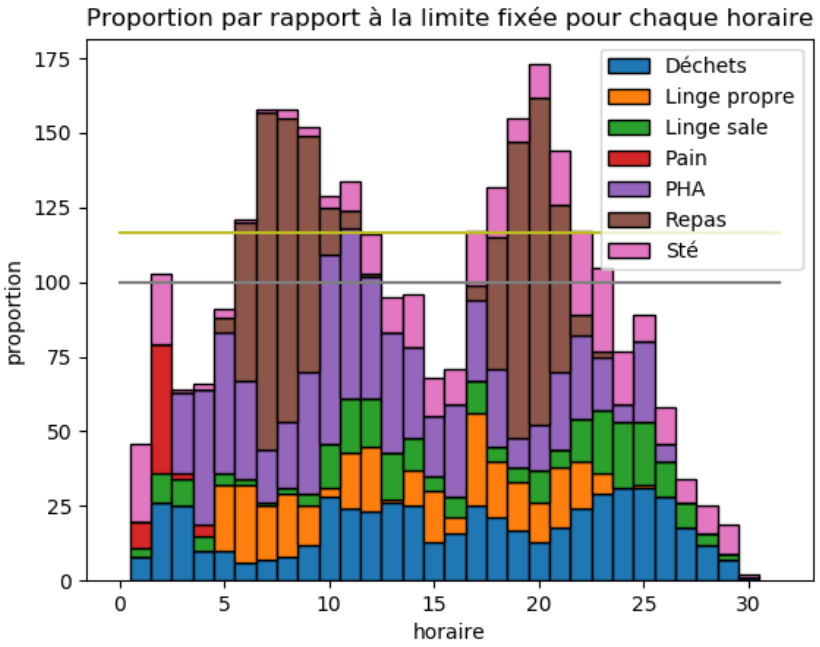


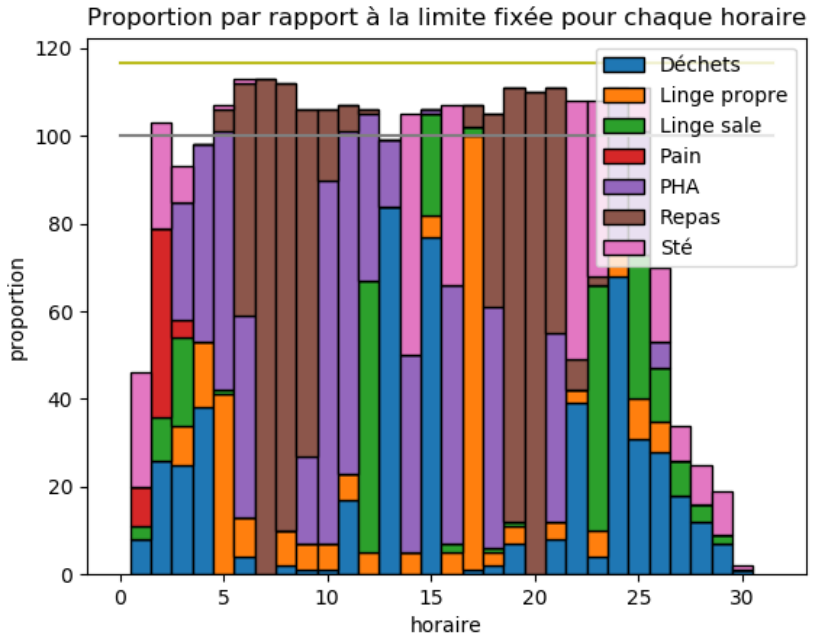


- Respect de l'allure des statistiques fournies
- Emploi du temps clairement équilibré

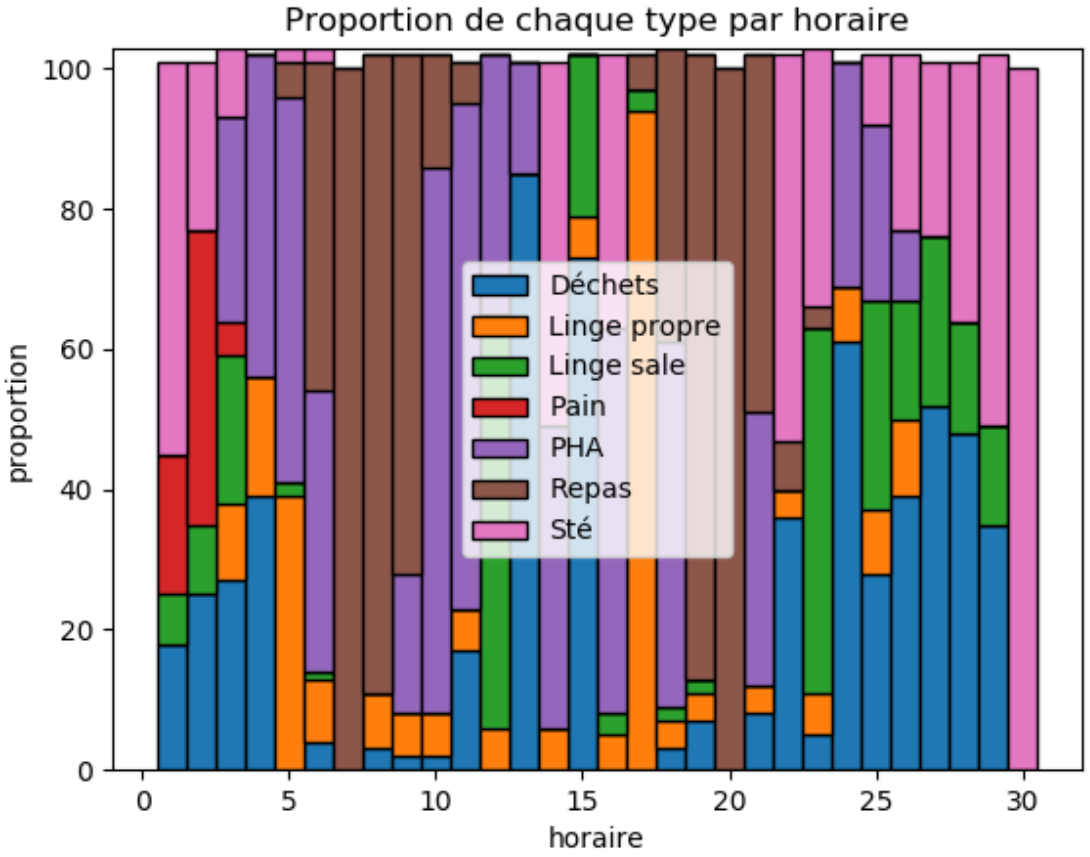
Bilan avec l'entreprise

- Résultat concluant
- Test sur des stats plus larges que l'exemple





➢ Résultat sur des statistiques plus précises satisfaisant également



Pertes de conteneurs	
Déchets	5%
Linge propre	8%
Linge sale	10%
Pain	2%
PHA	2%
Repas	0%
Sté	8%

Annexe - Code

```
import xlrd
import numpy as np
from matplotlib import pyplot
from random import *
from math import *

# ouverture du fichier Excel

wb = xlrd.open_workbook('C:/Users/aymer/Documents/TIPE/python_scripts/Classeur1.xls')

# feuilles dans le classeur

u = wb.sheet_names()

# récupération des données et prépa de l'edt

sh = wb.sheet_by_name(u'Feuil1')
h=len(sh.row_values(0))
l=len(sh.col_values(0))

def edt(h,l):
    edt=np.array(h*[l*['00000000000000000000']])
    return(edt)
```

```
EDT=edt((h+1),l)

for colnum in range(0,h):
    colonneh = sh.col_values(colnum)
    EDT[colnum,0]=str(colonneh[0])
    EDT[h,0]='horaires'
    heure=1
    for rownum in range(1,l):
        EDT[h,rownum]=str(heure)+'h'
        heure+=1

#saisie des contraintes

print(sh.row_values(0))
print('choisissez les types de contraintes ou tapez 0')
a1=[]
a1t=input()
while a1t!='0':
    if a1t in (sh.row_values(0)):
        a1+=[a1t]
        print('choisissez une autre contrainte ou tapez 0')
        a1t=input()
    else:
        print('saisie incorrecte , veuillez réessayer')
        a1t=input()

for i in range(h):
    for rownum in range(1,l):
        cell = sh.cell(rownum,i)
        EDT[i,rownum]=cell.value
```

```
#calcul de la limite par horaire

limiteh = 0
for colnum in range(0,h):
   colonneh = sh.col_values(colnum)
   for rownum in range(1,l):
      cell = sh.cell(rownum,colnum)
      limiteh+= cell.value
limiteh=int((limiteh/(l-1))+1)
epsilonlim=int(((10/60)*limiteh))

#représentation graphique EDT

def représentation():
   listex=[]
   for i in range (h):
      xi=[]
      for horaire in range(1,l):
         prop=int(100*(float(EDT[i,horaire])/limiteh))
         xi+=prop*[horaire]
      listex+=[xi]
   possib = 100*(limiteh + epsilonlim)/limiteh
   bins = [x + 0.5 for x in range(0, l)]
   pyplot.hist(listex, bins = bins, color = None,
            edgecolor = 'black', label = sh.row_values(0),
            histtype = 'barstacked')
   pyplot.ylabel('proportion')
   pyplot.xlabel('horaire')
   pyplot.title('EDT')
   pyplot.plot([0,l+0.5],[100,100])
   pyplot.plot([0,l+0.5],[possib,possib])
   pyplot.legend()
   pyplot.show()
représentation()
```

```
#fct utiles

def sumledt(colnum):
    s=0
    for i in range(1,l):
        s+=float(EDT[colnum,i])
    return(s)

def sumcoledt(rownum):
    s=0
    for i in range(h):
        s+=float(EDT[i,rownum])
    return(s)

#repérage des paquets

def repérage_paquets():
    paquets=[]
    num=-1
    dep=False
    for i in range (1,l):
        s=sumcoledt(i)
        if s>limiteh+epsilonlim:
            if not dep:
                num+=1
                paquets.append([i,i,0])
                dep = True
            else:
                paquets[num][1]=i
        else:
            dep = False
    for k in range(len(paquets)):
        paquets[k][2]=(paquets[k][0]+paquets[k][1])//2
    return(paquets)
```

```
#distribution et égalisation

def tri_adapt(t):
    for k in range(1,len(t)):
        temp=t[k]
        j=k
        while j>0 and float(temp[0])<float(t[j-1][0]):
            t[j]=t[j-1]
            j-=1
        t[j]=temp
    return t

def recup2(paquets,k):
    colonne=[]
    hor=(paquets[k][2])
    for i in range (h):
        colonne.append([EDT[i][hor],EDT[i][0]])
    tri_adapt(colonne)
    return colonne
```

```
def partition2(paquets,k):
    rcp=recup2(paquets,k)
    numh=paquets[k][2]
    typc=np.array(EDT.take([0],axis=1)).tolist()
    for i in range(len(rcp)):
        if not (rcp[i][1]) in a1:
            dps=sumcoledt(numh)-(limiteh)
            if dps>0:
                q=float(rcp[i][0])
                num=typc.index([rcp[i][1]])
                if dps>=q:
                    EDT[num][numh]='0.0'
                    if numh==0 :
                        EDT[num][numh+1]=str(float(EDT[num][numh+1])+q)
                    if numh== (l-1) :
                        EDT[num][numh-1]=str(float(EDT[num][numh-1])+q)
                    else :
                        EDT[num][numh-1]=str(float(EDT[num][numh-1])+ceil(q//2))
                        EDT[num][numh+1]=str(float(EDT[num][numh+1])+ceil(q//2))
                if dps<q:
                    EDT[num][numh]=str(q-dps)
                    if numh==0 :
                        EDT[num][numh+1]=str(float(EDT[num][numh+1])+(dps))
                    if numh== (l-1) :
                        EDT[num][numh-1]=str(float(EDT[num][numh-1])+(dps))
                    else:
                        EDT[num][numh-1]=str(float(EDT[num][numh-1])+ceil((dps)//2))
                        EDT[num][numh+1]=str(float(EDT[num][numh+1])+ceil((dps)//2))
```

```
def partition(paquets):
    for k in range(len(paquets)):
        partition2(paquets,k)

def iteration():
    partition(repérage_paquets())
f=False
while f == False:
    iteration()
    f=True
    for i in range(1,l):
        if sumcoledt(i)>(limiteh+epsilonlim):
            f = False

représentation()

#emploi du temps graphique, création + simplification

EDTS=edt((h+1),l)
for colnum in range(h):
    colonneh = sh.col_values(colnum)
    EDTS[colnum,0]=str(colonneh[0])
    EDTS[h,0]='horaires'
    heure=1
    for rownum in range(1,l):
        EDTS[h,rownum]=str(heure)+'h'
        heure+=1
```

```
ls=[]
for i in range(1,l):
   colonne=[]
   s2=0
   for k in range (h):
      s=sumledt(k)
      if float(EDT[k][i]) <= (a/100)*s: // où (a/100) est le pourcentage de  précision dépendant de la précision horaire
         colonne.append([EDT[k][0],'0'])
      else:
         colonne.append([EDT[k][0],EDT[k][i]])
         s2+=float(EDT[k][i])
   for k in range(h):
      if s2!=0:
         EDTS[k,i]=str(ceil(float(colonne[k][1])*100/s2))
   ls.append(s2)

#résidus

def card(colnum):
   cardlist=sh.col_values(colnum)
   card=0
   for i in range(1,len(cardlist)):
      cell = sh.cell(i,colnum)
      card+=cell.value
   return(card)

reste=np.array(h*[2*['00000000000000000']])

def sumledtg(colnum):
   s=0
   for i in range(1,l):
      s+=float(EDTS[colnum,i])*ls[i-1]/100
   return(s)
```

```
def rest(reste):
   for colnum in range(h):
      colonneh = sh.col_values(colnum)
      reste[colnum,0]=str(colonneh[0])
      cardi=card(colnum)
      diff=cardi-sumledtg(colnum)
      if diff<=0:
         reste[colnum,1]=0
      else:
         reste[colnum,1]=int(diff*100/cardi)

rest(reste)
print(reste)

#représentation graphique EDTS

def représentationS():
   listex=[]
   for i in range (h):
      xi=[]
      for horaire in range(1,l):
         prop=int(float(EDTS[i,horaire]))
         xi+=prop*[horaire]
      listex+=[xi]
   bins = [x + 0.5 for x in range(0, l)]
   pyplot.hist(listex, bins = bins, color = None,
            edgecolor = 'black', label = sh.row_values(0),
            histtype = 'barstacked')
   pyplot.ylabel('proportion')
   pyplot.xlabel('horaire')
   pyplot.title('EDT')
   pyplot.legend()
   pyplot.show()

représentationS()
```